UNITED STATES PATENT AND TRADEMARK OFFICE

𝒯𝐻

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/787,005	02/25/2004	Takeshi Ogasawara	JP920030021US1	3932

7590   04/13/2007

LOUIS P. HERZBERG
IBM Corporation
Intellectual Property Law Dept.
P.O. Box 218
Yorktown Heights, NY 10598

EXAMINER
BODDEN, EVRAL E

ART UNIT	PAPER NUMBER
2109	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	04/13/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

PTOL-90A (Rev. 10/06)

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). ·
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) ☒ Responsive to communication(s) filed on *2/25/04*.
2a) ☐ This action is **FINAL**.   2b) ☒ This action is non-final.
3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) ☒ Claim(s) *1-22* is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5) ☐ Claim(s) _____ is/are allowed.
6) ☒ Claim(s) *1-22* is/are rejected.
7) ☐ Claim(s) _____ is/are objected to.
8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) ☒ The specification is objected to by the Examiner. ·
10) ☒ The drawing(s) filed on *2/25/04* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☒ All   b)☐ Some * c)☐ None of:

      1.☒ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
    Paper No(s)/Mail Date *10/09/06, 12/10/04*.

4) ☐ Interview Summary (PTO-413)
    Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____ .

DETAILED ACTION

1.   This action is in response to the following communication: Non-provisional application filed

02/25/2004.

2.   Claims 1-22 are pending. Claims 1, 4, 5, 11, 12, 13, and 19 are independent claims. Claims

2, 3, 6-10, 14-18, 20-22 are dependent claims.

Claim Rejections - 35 USC § 101

3.   35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new
>
> and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4.   **Claims 1-4, 11-12, 14-15 and 18-22** are rejected under 35 U.S.C. 101 because the claimed

invention is directed to non-statutory subject matter.

Claims 1-4, 11-12, 14-15 and 18-22 are directed to a compiler or compiling.

This claimed subject matter lacks a practical application of a judicial exception (law of nature,

abstract idea, naturally occurring phenomenon) since it fails to produce a useful, concrete and

tangible result.

Specifically, the claimed subject matter does not produce a tangible result because the

claimed subject matter fails to produce a result that is limited to having real world value rather than a

result that may be interpreted to be abstract in nature as, for example, a thought, a computation, or

manipulated data. More specifically, the claimed subject matter provides for "generating" (claims 1, 4,

11, 12, 14, 15, 16, 18, 19, 21 and 22) and/or "detecting (claims 2, 3, 20). These produced results

remain in the abstract and, thus, fails to achieve the required status of having real world value.

5.   **Claims 5-10, 13 and 17** are rejected under 35 U.S.C. 101 because the claimed invention is

directed to non-statutory subject matter.

   Claim(s) 5-10, 13 and 17 are directed to a compiler or compiling.

   This claimed subject matter lacks a practical application of a judicial exception (law of nature,

abstract idea, naturally occurring phenomenon) since it fails to produce a useful, concrete and

tangible result.

   Specifically, the claimed subject matter does not produce a useful result because the claimed

subject matter fails to sufficiently reflect at least one practical utility set forth in the descriptive portion

of the specification. More specifically, while the described practical utility (utilities) is (are) directed to

optimizing code, the claimed subject matter relates ONLY to conversion of immutable string variables.

Since an immutable variable is defined as a variable that cannot be changed, it would not be useful to

concatenate such a variable.

6.   **Claims 1-14** are rejected under 35 U.S.C. 101 because the claimed invention is directed to

non-statutory subject matter.

   The claims lack the necessary physical articles or objects to constitute a machine or a

manufacture within the meaning of 35 USC 101. They are clearly not a series of steps or acts to be a

process nor are they a combination of chemical compounds to be a composition of matter. As such,

they fail to fall within a statutory category. They are, at best, functional descriptive material *per se*.

The Examiner notes that although claims 1-10 include "device" in the preamble, the detection,

generation, and elimination units amount to nothing more than descriptions of software.

   Descriptive material can be characterized as either "functional descriptive material" or "non-

functional descriptive material." Both types of "descriptive material" are non-statutory when claimed

as descriptive material *per se*, 33 F.3d at 1360, 31 USPQ2d at 1759. When <u>functional</u> descriptive

material is recorded on some computer-readable medium, it becomes structurally and functionally

interrelated to the medium and will be statutory in most cases since use of technology permits the

function of the descriptive material to be realized. Compare *In re Lowry*, 32 F.3d 1579, 1583-84, 32

USPQ2d 1031, 1035 (Fed. Cir. 1994)

Merely claiming non-functional descriptive material, i.e., abstract ideas, stored on a computer-

readable medium, in a computer, or on an electromagnetic carrier signal, does not make it statutory.

See *Diehr*, 450 U.S. at 185-86, 209 USPQ at 8 (noting that the claims for an algorithm in *Benson*

were unpatentable as abstract ideas because "[t]he sole practical application of the algorithm was in

connection with the programming of a general purpose computer.").

Claims 1–15, and 17– 22 are also rejected under 35 U.S.C. 112, first paragraph.

Specifically, since the claimed invention is not supported by either a substantial, credible,

and concrete asserted utility or a well established utility for the reasons set forth above,

one skilled in the art clearly would not know how to use the claimed invention.

Claim Rejections - 35 USC § 112

7.   The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

$5 - 10$

Claim 5 and 9 are also rejected under 35 U.S.C. 112, first paragraph, as failing to comply with

the enablement requirement. The claim(s) contains subject matter which was not described in the

specification in such a way as to enable one skilled in the art to which it pertains, or with which it is

most nearly connected, to make and/or use the invention. Specifically, due to the level of abstraction

of the following portion of claim 5:

- an instruction elimination unit for eliminating the immutable-to-mutable conversion

   instruction and for causing the mutable string variable as a source variable of the

   mutable-to-immutable conversion instruction, to be used as the mutable string variable

   obtained from conversion by the immutable-to-mutable conversion instruction after the

   immutable-to-mutable conversion instruction, if an instruction to be executed between

   the mutable-to-immutable conversion instruction and the immutable-to-mutable

   conversion instruction does not modify a character string stored in the mutable string

   variable as the source variable of the mutable-to-immutable conversion instruction, and

   if an instruction to be executed between the immutable-to-mutable conversion

   instruction and use of the mutable string variable obtained from the conversion by the

   immutable-to-mutable conversion instruction does not modify any of the mutable string

   variable as the source variable of the mutable-to-immutable conversion instruction and

   the mutable string variable obtained from the conversion by the immutable-to-mutable

   conversion instruction.

in addition to the level of abstraction of the following portion of claim 9:

- a partial redundancy elimination unit for executing a partial redundancy elimination

  process of moving the immutable-to-mutable conversion instruction detected by the

  immutable-to-mutable conversion instruction detection unit to each control flow edge

  which merges into a single control flow before the immutable-to-mutable conversion

  instruction, wherein the instruction elimination unit eliminates the immutable-to-mutable

  conversion instruction, if an instruction to be executed between the mutable-to-

  immutable conversion instruction and the immutable-to-mutable conversion instruction

  does not modify a character string stored in the mutable string variable as the source

  variable of the mutable-to-immutable conversion instruction and if an instruction to be

  executed between the immutable-to-mutable conversion instruction and the use of the

  mutable string variable obtained from the conversion by the immutable-to-mutable

  conversion instruction does not modify any of the mutable string variable as the source

  variable of the mutable-to-immutable conversion instruction and the mutable string

  variable obtained from the conversion by the immutable-to-mutable conversion

  instruction, in a program obtained after the partial redundancy elimination process has

  been executed.

a person skilled in the art to which it pertains to, would encounter difficulties of enablement, based on

these descriptions.

8.   *Specification*

The abstract of the disclosure is objected to because of the following informalities:

A: Second sentence, "manipulates a character string includes append

instruction....", is grammatically incorrect, examiner suggest using,

"manipulates a character string, including an append instruction...".

The disclosure is objected to because of the following informalities:

A: Grammatical error on (Page 2, line 16), "the operation efficiency of

the server has been lowered", examiner suggest using "the

operation efficiency of the server is lowered".

B: Spelling incorrection, (Page 4, line 16), "source cord" should be

"source code".

                              Claim Objections

Claim 14 is objected to because of the following informality:

A: Claim 14, Line1, references "recording medium having any a

compiler program". Remove the word "any".

Appropriate correction is required.

Claim Rejections - 35 USC § 102

9.   The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis

for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 1 and **3,** are rejected under 35 U.S.C. 102(e) as being unpatentable

over Arnold et al. (hereinafter Arnold) 6,523,168 issued Feb. 18, 2003, and filed Jul. 1, 1999.

In regards to **independent claim 1,** Arnold teaches:

- a compiler device for optimizing a program which manipulates a character string

Due to the reference to compiler in (column 5, lines 37 – 38):

- *A translation program consistent with the invention may incorporate a compiler to generate suitable program code.*

and the reference to string concatenation in (column 7, 63-64):

- *Reduction of Object Creation During String Concatenation*

, it's inherent that the optimization of a compiler is achieved by targeting strings.

In regards to **independent claim 1,** Arnold teaches the compiler device comprising:

- an append instruction detection unit for detecting an append instruction to append a

   character string to a string variable for storing a character string, in the program;

Due to the reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation:*

,it's inherent that character stings concatenations are detected.

In regards to **independent claim 1,** Arnold teaches a:

- a store code generation unit for generating, as a substitute for each of a plurality of the

  append instructions detected by the append instruction detection unit, a store code for

  storing data of an appendant character string to be appended to the string variable by

  the append instruction into a buffer, the plurality of append instructions appending the

  character strings to the same string variable;

Due to the reference to generating program code for each of a plurality of operation in Arnold

on (column 12 lines 45 –57):

> *(a) generating program code for the second computer program that allocates a reusable temporary object for handling multiple operations in the second computer program that require the use of temporary storage; and (b) for each of a plurality of operations defined in the first computer program that require the use of temporary storage, generating program code in the second computer program to utilize the reusable temporary object in performing the operation.*

and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation.*

, it's inherent store code/program code for the plurality of operations/appended instructions is being

generated.

In regards to **independent claim 1,** Arnold teaches a:

- an append code generation unit for generating an append code for appending a plurality

  of the appendant character strings to the string variable, at a position to be executed

  before an instruction to refer to the string variable in the program.

Due to the reference to reduction of multiple sting concatenation in Arnold on (column 4, lines

19-28):

- *rather than creating a new mutable string object (as well as an underlying character array object) for each string concatenation operation, an existing mutable string object, allocated at the initialization of a program (or a thread thereof), is used as the temporary storage for each operation. The total number of objects created as a result of multiple string concatenation operations is therefore reduced.*

it's inherent that a plurality of appendant character strings are stored.

In regards to **dependent claim 3**, Arnold teaches a:

- The compiler device according to claim 1, wherein the append instruction detection unit detects, as the append instruction, a combination of: an instruction to convert an immutable string variable in which a process of appending a character string is not allowed, into a mutable string variable in which a process of appending a character string is allowed; an instruction to append the appendant character string to the mutable string variable; and an instruction to convert the mutable string variable into the immutable string variable.

Due to the reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

, and the reference to plurality of string concatenation operation in Arnold on (column 12, lines 45-57):

- *translating a first computer program into a second computer program, the method comprising: (a) generating program code for the second computer program that allocates a reusable temporary object for handling multiple operations in the second computer program that require the use of temporary storage; and (b) for each of a plurality of operations defined in the first computer program that require the use of temporary storage, generating program code in the second computer program to utilize the reusable temporary object in performing the operation.*
- *wherein each of the plurality of operations includes a string concatenation operation.*

and the reference to includes mutable string object in Arnold on (column 14, lines 45-46) :

- *wherein the reusable temporary object includes a mutable string object.*

, it's inherrent that append instructions applicable to mutable character strings are being detected,

and optimized.

10.　　　　　　　　　*Claim Rejections - 35 USC § 103*

1.　　　The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2.　　　The factual inquiries set forth in *Graham* v. *John Deere Co.*, 383 U.S. 1, 148 USPQ 459

(1966), that are applied for establishing a background for determining obviousness under 35

U.S.C. 103(a) are summarized as follows:

1.　　Determining the scope and contents of the prior art.
2.　　Ascertaining the differences between the prior art and the claims at issue.
3.　　Resolving the level of ordinary skill in the pertinent art.
4.　　Considering objective evidence present in the application indicating obviousness or nonobviousness.

This application currently names joint inventors. In considering patentability of the claims

under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was

commonly owned at the time any inventions covered therein were made absent any evidence to the

contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and

invention dates of each claim that was not commonly owned at the time a later invention was made in

order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35

U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Claims 2, 4-22, are rejected under 35 U.S.C. 103(a) as being unpatentable over Arnold et al.

(hereinafter Arnold) 6,523,168 issued Feb. 18, 2003, and filed Jul. 1, 1999, in view of Bates et al.

(hereinafter Bates) 2003/0145312A1 published Jul. 31, 2003, and filed Jan. 30, 2002:

In regards to **independent claim 4,** Arnold teaches:

- A compiler device for optimizing a program which manipulates a character string,

The reference to compiler in (column 5 lines 37 – 38):

- *A translation program consistent with the invention may incorporate a compiler to generate suitable program code.*

and the reference to string concatenation in (column 7, 63-64):

- *Reduction of Object Creation During String Concatenation*

implies that the optimization of a compiler is achieved by targeting character strings.

In regards to **independent claim 4,** Arnold teaches:

- the compiler device comprising: an append instruction detection unit for detecting an append instruction to append a character string to a string variable for storing a character string, in the program;

The reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation.*

, implies that character stings concatenations are detected.

In regards to **independent claim 4,** Arnold teaches:

- a store code generation unit for generating, as a substitute for each of a plurality of the append instructions detected by the append instruction detection unit, a store code for storing an address in memory where an appendant character string to be appended to the string variable by the append instruction is stored, into a buffer, the plurality of append instructions appending character strings to the same string variable;

The reference to generating program code for each of a plurality of operation in Arnold

on (column 12 lines 45 –57):

> *(a) generating program code for the second computer program that allocates a reusable temporary object for handling multiple operations in the second computer program that require the use of temporary storage; and (b) for each of a plurality of operations defined in the first computer program that require the use of temporary storage, generating program code in the second computer program to utilize the reusable temporary object in performing the operation.*

and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation.*

, implies that store code/program code for the plurality of operations/appended instructions is being

generated.

In regards to **independent claim 4,** Arnold teaches:

- an append code generation unit for generating an append code for appending a plurality

   of the appendant character strings stored in a plurality of the addresses, to the string

   variable, at a position to be executed before an instruction to refer to the string variable

   in the program.

The reference to reduction of multiple sting concatenation in Arnold on (column 4, lines 19-28):

> *rather than creating a new mutable string object (as well as an underlying character array object) for each string concatenation operation, an existing mutable string object, allocated at the initialization of a program (or a thread thereof), is used as the temporary storage for each operation. The total number of objects created as a result of multiple string concatenation operations is therefore reduced.*

implies that a plurality of appendant character strings are stored.

In regards to **independent claim 5,** Arnold teaches:

- A compiler device for optimizing a program which manipulates a character string, the

   compiler device comprising: a mutable-to-immutable conversion instruction detection

   unit for detecting a mutable-to-immutable conversion instruction to convert a mutable

string variable in which a process of appending a character string is allowed, into an

immutable string variable in which a process of appending a character string is not

allowed; an immutable-to-mutable conversion instruction detection unit for detecting an

immutable-to-mutable conversion instruction to convert the immutable string variable

into the mutable string variable; and an instruction elimination unit for eliminating the

immutable-to-mutable conversion instruction and for causing the mutable string variable

as a source variable of the mutable-to-immutable conversion instruction, to be used as

the mutable string variable obtained from conversion by the immutable-to-mutable

conversion instruction after the immutable-to-mutable conversion instruction, if an

instruction to be executed between the mutable-to-immutable conversion instruction and

the immutable-to-mutable conversion instruction does not modify a character string

stored in the mutable string variable as the source variable of the mutable-to-immutable

conversion instruction, and if an instruction to be executed between the immutable-to-

mutable conversion instruction and use of the mutable string variable obtained from the

conversion by the immutable-to-mutable conversion instruction does not modify any of

the mutable string variable as the source variable of the mutable-to-immutable

conversion instruction and the mutable string variable obtained from the conversion by

the immutable-to-mutable conversion instruction.

The reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

, the reduction of multiple sting concatenation in Arnold on (column 4,

lines 19-28):

- *rather than creating a new mutable string object (as well as an underlying character array object) for each string concatenation operation, an existing mutable string object, allocated at the initialization of a program (or a thread thereof), is used as the temporary storage for each operation. The total number of objects created as a result of multiple string concatenation operations is therefore reduced, easing allocation and collection overhead, and accordingly improving overall system performance.*

and in Arnold on (column 12 lines 45 –57):

- *translating a first computer program into a second computer program, the method comprising: (a) generating program code for the second computer program that allocates a reusable temporary object for handling multiple operations in the second computer program that require the use of temporary storage; and (b) for each of a plurality of operations defined in the first computer program that require the use of temporary storage, generating program code in the second computer program to utilize the reusable temporary object in performing the operation.*

implies that character strings concatenations are detected, stored, and optimized via the reduction of

multiple sting concatenations.

In addition, the reference to the translation process being utilized using

any number of representations in Arnold on (column 5, line 65-67):

- *being translated, as well as the computer program being generated as a result of the translation process, may be utilized using any number of representations, whether human or machine readable in nature.*

implies that various methods are utilized to detect, append, and optimize various types of character

strings.

In regards to **claim 6**, Arnold teaches:

- The compiler device according to claim 5, wherein the instruction elimination unit further

  eliminates the mutable-to-immutable conversion instruction if a character string stored in

  the immutable string variable is not referred to.

The reference to deallocating in Arnold on (column 1, Lines 26- 31)

- *some mechanism for removing, or "deallocating", unused objects is also provided, typically either through the use of specific program instructions or through an automated process known as garbage collection.*

implies that strings not referenced are deallocated and eliminated.

In regards to **claim 8**, Arnold teaches:

- The compiler device according to claim 5, wherein the immutable-to-mutable conversion instruction detection unit detects, as the immutable-to-mutable conversion instruction, a combination of: an instruction to reserve a memory area to be used as a mutable string variable; and an instruction to append a character string stored in the immutable string variable to the mutable string variable.

, the reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

and the reference to allocating memory for string concatenation operation in Arnold in (column 14, line 15-23):

- *the use of temporary storage, to generate program code in the second computer program to utilize the reusable temporary object in performing the operation.*

- *The apparatus of claim 17, wherein each of the plurality of operations includes a string concatenation operation.*

implies that memory is being reserved for the purpose of optimizing character strings.

In regards to **claim 9,** Arnold teaches:

- The compiler device according to claim 5, further comprising: a partial redundancy elimination unit for executing a partial redundancy elimination process of moving the immutable-to-mutable conversion instruction detected by the immutable-to-mutable conversion instruction detection unit to each control flow edge which merges into a single control flow before the immutable-to-mutable conversion instruction, wherein the instruction elimination unit eliminates the immutable-to-mutable conversion instruction, if an instruction to be executed between the mutable-to-immutable conversion instruction and the immutable-to-mutable conversion instruction does not modify a character string

stored in the mutable string variable as the source variable of the mutable-to-immutable

conversion instruction and if an instruction to be executed between the immutable-to-

mutable conversion instruction and the use of the mutable string variable obtained from

the conversion by the immutable-to-mutable conversion instruction does not modify any

of the mutable string variable as the source variable of the mutable-to-immutable

conversion instruction and the mutable string variable obtained from the conversion by

the immutable-to-mutable conversion instruction, in a program obtained after the partial

redundancy elimination process has been executed.

, the reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

, the reference to plurality of string concatenation operation in Arnold on (column 12, lines 45-57):

- *translating a first computer program into a second computer program, the method comprising:*
  (a) generating program code for the second computer program that allocates a reusable
  temporary object for handling multiple operations in the second computer program that require
  the use of temporary storage; and (b) for each of a plurality of operations defined in the first
  computer program that require the use of temporary storage, generating program code in the
  second computer program to utilize the reusable temporary object in performing the operation.

and the reference to includes mutable string object in Arnold on (column 14, lines 45-46):

- *wherein each of the plurality of operations includes a string concatenation operation.*

implies that append instructions applicable to mutable character strings are being detected, and

optimized.

In regards to **independent claim 11,** Arnold teaches:

- a compiler device for optimizing a program which manipulates a character string

The reference to compiler in (column 5 lines 37 – 38):

- *A translation program consistent with the invention may incorporate a compiler to generate*
  suitable program code.

and the reference to string concatenation in (column 7, 63-64):

- *Reduction of Object Creation During String Concatenation*

, implies that the optimization of a compiler is achieved by targeting character strings.

In regards to **independent claim 11,** Arnold teaches the compiler program causing the computer to function as:

- an append instruction detection unit for detecting an append instruction to append a character string to a string variable for storing a character string, in the program;

The reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation.*

implies that character stings concatenations are detected.

In regards to **independent claim 11,** Arnold teaches:

- a store code generation unit for generating, as a substitute for each of a plurality of the append instructions detected by the append instruction detection unit, a store code for storing data of an appendant character string to be appended to the string variable by the append instruction into a buffer, the plurality of append instructions appending the character strings to the same string variable; and

The reference to generating program code for each of a plurality of operation in Arnold on (column 12 lines 45 –57):

> *(a) generating program code for the second computer program that allocates a reusable temporary object for handling multiple operations in the second computer program that require the use of temporary storage; and (b) for each of a plurality of operations defined in the first computer program that require the use of temporary storage, generating program code in the second computer program to utilize the reusable temporary object in performing the operation.*

and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation.*

, implies that store code/program code for the plurality of operations/appended instructions is being

generated.

In regards to **independent claim 11,** Arnold teaches:

- an append code generation unit for generating an append code for appending a plurality

  of the appendant character strings to the string variable, at a position to be executed

  before an instruction to refer to the string variable in the program.

The reference to reduction of multiple sting concatenation in Arnold on (column 4, lines 19-28):

- *rather than creating a new mutable string object (as well as an underlying character array object) for each string concatenation operation, an existing mutable string object, allocated at the initialization of a program (or a thread thereof), is used as the temporary storage for each operation. The total number of objects created as a result of multiple string concatenation operations is therefore reduced.*

implies that a plurality of appendant character strings are stored.

In regards to **independent claim 12,** Arnold teaches:

- A compiler program for optimizing a program which manipulates a character string,

The reference to compiler in (column 5 lines 37 – 38):

- *A translation program consistent with the invention may incorporate a compiler to generate suitable program code.*

and the reference to string concatenation in (column 7, 63-64):

- *Reduction of Object Creation During String Concatenation*

, implies that the optimization of a compiler is achieved by targeting character strings.

In regards to **independent claim 12,** Arnold teaches by using a computer, the compiler

program causing the computer to function as:

- an append instruction detection unit for detecting an append instruction to append a

    character string to a string variable for storing a character string, in the program;

The reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

, and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation.*

, implies that character stings concatenations are detected.

In regards to **independent claim 12,** Arnold teaches:

- a store code generation unit for generating, as a substitute for each of a plurality of the

    append instructions detected by the append instruction detection unit, a store code for

    storing an address in memory where an appendant character string to be appended to

    the string variable by the append instruction is stored, into a buffer, the plurality of

    append instructions appending the character strings to the same string variable;

The reference to generating program code for each of a plurality of operation in Arnold

on (column 12 lines 45 –57):

> (a) *generating program code for the second computer program that allocates a reusable temporary*
> *object for handling multiple operations in the second computer program that require the use of*
> *temporary storage; and (b) for each of a plurality of operations defined in the first computer*
> *program that require the use of temporary storage, generating program code in the second*
> *computer program to utilize the reusable temporary object in performing the operation.*

, and in Arnold on (column 12, line 56)

- *wherein each of the plurality of operations includes a string concatenation operation.*

, implies that store code/program code for the plurality of operations/appended instructions is being

generated.

In regards to **independent claim 12,** Arnold teaches:

- an append code generation unit for generating an append code for appending a plurality

  of the appendant character strings stored in a plurality of the addresses, to the string

  variable, at a position to be executed before an instruction to refer to the string variable

  in the program.

The reference to reduction of multiple sting concatenation in Arnold on (column 4, lines 19-28):

- *rather than creating a new mutable string object (as well as an underlying character array*
  object) for each string concatenation operation, an existing mutable string object, allocated at
  the initialization of a program (or a thread thereof), is used as the temporary storage for each
  operation. The total number of objects created as a result of multiple string concatenation
  operations is therefore reduced.

implies that a plurality of appendant character strings are stored.

In regards to **independent claim 13,** Arnold teaches:

- A compiler program for optimizing a program which manipulates a character string, by

  using a computer, the compiler program causing the computer to function as: a

  mutable-to-immutable conversion instruction detection unit for detecting a mutable-to-

  immutable conversion instruction to convert a mutable string variable in which a process

  of appending a character string is allowed, into an immutable string variable in which a

  process of appending a character string is not allowed; an immutable-to-mutable

  conversion instruction detection unit for detecting an immutable-to-mutable conversion

  instruction to convert the immutable string variable into the mutable string variable; and

  an instruction elimination unit for eliminating the immutable-to-mutable conversion

instruction and for causing the mutable string variable as a source variable of the

mutable-to-immutable conversion instruction, to be used as the mutable string variable

obtained from conversion by the immutable-to-mutable conversion instruction after the

immutable-to-mutable conversion instruction, if an instruction to be executed between

the mutable-to-immutable conversion instruction and the immutable-to-mutable

conversion instruction does not modify a character string stored in the mutable string

variable as the source variable of the mutable-to-immutable conversion instruction and if

an instruction to be executed between the immutable-to-mutable conversion instruction

and use of the mutable string variable obtained from the conversion by the immutable-

to-mutable conversion instruction does not modify any of the mutable string variable as

the source variable of the mutable-to-immutable conversion instruction and the mutable

string variable obtained from the conversion by the immutable-to-mutable conversion

instruction.

, the reference to detecting a conflicting operation in Arnold on (column 13, line 22):

- *detecting a conflicting operation in the first computer*

, the reference to plurality of string concatenation operation in Arnold on (column 12, lines 45-57):

- *translating a first computer program into a second computer program, the method comprising: (a) generating program code for the second computer program that allocates a reusable temporary object for handling multiple operations in the second computer program that require the use of temporary storage; and (b) for each of a plurality of operations defined in the first computer program that require the use of temporary storage, generating program code in the second computer program to utilize the reusable temporary object in performing the operation.*
- *wherein each of the plurality of operations includes a string concatenation operation.*

and the reference to includes mutable string object in Arnold on (column 14, lines 45-46):

- *wherein the reusable temporary object includes a mutable string object.*

implies that append instructions applicable to mutable character strings are being detected, and

optimized.

In regards to **claim 14, 15, 16, 17, and 18** Arnold teaches:

- A recording medium having any a compiler program according to claim 11 recorded

  thereon.

The reference in Fig.1 and Fig. 2 to a computer system in Arnold on (column 4, line 39-43)

- *FIG. 1 is a block diagram of a networked computer system consistent with the invention.*
- *FIG. 2 is a block diagram of an exemplary hardware and software environment for a computer from the networked computer system of FIG. 1.*

implies that a "recording medium" or "computer program product" is being utilized to record and

effectively execute specified functions of the compiler program.

In regards to **independent claim 19**, Arnold teaches:

- A method for optimizing a program which manipulates a character string,

The reference to compiler in (column 5 lines 37 – 38):

- *A translation program consistent with the invention may incorporate a compiler to generate suitable program code.*

and the reference to string concatenation in (column 7, 63-64):

- *Reduction of Object Creation During String Concatenation*

, implies that the optimization of a compiler is achieved by targeting character strings.

In regards to **independent claim 19**, Arnold teaches:

- the method comprising: detecting an append instruction to append a character string to

  a string variable for storing a character string,

The reference to detecting a conflicting operation in Arnold on (column 13, line 22):

> • *detecting a conflicting operation in the first computer*

and in Arnold on (column 12, line 56)

> • *wherein each of the plurality of operations includes a string concatenation operation.*

implies that character stings concatenations are detected.

In regards to **independent claim 19**, Arnold teaches:

- generating, as a substitute for each of a plurality of the append instructions detected by the append instruction detection unit, a store code for storing data of an appendant character string to be appended to the string variable by the append instruction into a buffer, the plurality of append instructions appending the character strings to the same string variable;

The reference to generating program code for each of a plurality of operation in Arnold on (column 12 lines 45 –57):

> *(a) generating program code for the second computer program that allocates a reusable temporary object for handling multiple operations in the second computer program that require the use of temporary storage; and (b) for each of a plurality of operations defined in the first computer program that require the use of temporary storage, generating program code in the second computer program to utilize the reusable temporary object in performing the operation.*

, and in Arnold on (column 12, line 56)

> • *wherein each of the plurality of operations includes a string concatenation operation.*

, implies that store code/program code for the plurality of operations/appended instructions is being generated.

In regards to **independent claim 19**, Arnold teaches:

- generating an append code for appending a plurality of the appendant character strings to the string variable, at a position to be executed before an instruction to refer to the string variable in the program.

The reference to reduction of multiple sting concatenation in Arnold on (column 4, lines 19-28):

- *rather than creating a new mutable string object (as well as an underlying character array object) for each string concatenation operation, an existing mutable string object, allocated at the initialization of a program (or a thread thereof), is used as the temporary storage for each operation. The total number of objects created as a result of multiple string concatenation operations is therefore reduced, easing allocation and collection overhead, and accordingly improving overall system performance.*

implies that a plurality of appendant character strings are stored.

In regards to **claim 21**, Arnold teaches:

- An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing optimization of a program which manipulates a character string, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 19.

The reference in Fig.1 and Fig.2 to a computer system in Arnold on (column 4, line 39-43)

- *FIG. 1 is a block diagram of a networked computer system consistent with the invention.*
- *FIG. 2 is a block diagram of an exemplary hardware and software environment for a computer from the networked computer system of FIG. 1.*

implies that an article of manufacture comprising a computer usable medium is being utilized to effect the steps of specified claims.

In regards to **claim 22**, Arnold teaches:

- A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for optimizing a program which manipulates a character string, said method steps comprising the steps of claim 19.

The reference in Fig.1 and Fig. 2 to a computer system in Arnold on (column 4, line 39-43)

- *FIG. 1 is a block diagram of a networked computer system consistent with the invention.*
- *FIG. 2 is a block diagram of an exemplary hardware and software environment for a computer from the networked computer system of FIG. 1.*

implies that a program storage device readable by machine is being utilized to effect the steps of the claim.

In regards to **claim 2 and 20,** Arnold does not appear to explicitly disclose:

- a reference instruction detection unit for detecting a reference instruction which first refers to the string variable after the character strings have been appended to the string variable by the plurality of append instructions, wherein the append code generation unit generates the append code at a position to be executed after the store codes and before the reference instruction.

However Bates discloses detecting instructions to be replaced (paragraph 8, line 1-10):

- *the invention provides a method of transforming source code, The method parses a source code statement. The method then determines if the source code statement includes a first operation that receives as an input a result of a second operation, where the second operation acts on a plurality of arguments. If the source code includes the first operation, then the method determines the order of the arguments, and then transforms the source code statement into a plurality of statements containing the first operation acting on one of the arguments.*

In regards to **claim 7 and 10,** Arnold does not appear to explicitly disclose:

- the instruction elimination unit moves the mutable-to-immutable conversion instruction to each branch destination of a branch instruction to be executed after the mutable-to-immutable conversion instruction, and executes partial dead assignment elimination for eliminating the mutable-to-immutable conversion instruction if a character string stored in the immutable string variable as a destination variable of the mutable-to-immutable conversion instruction is not referred to on each branch destination of the branch instruction.

However Bates discloses moving and elimination of code in (paragraph 7, line 1-10):

- *The method determines if a source code statement includes a first operation that receives input from a result of a second operation, where the second operation acts on a plurality of arguments. If the source code includes the first operation, then the method transforms the source code into a plurality of statements that include the first operation acting on one of the arguments.*

Arnold and Bates are analogous art because they are from the same field of endeavor, software optimization.

At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teaching of Arnold, and Bates before him or her, to modify the software of Arnold to include the changes of Bates because such modification would extend the optimization and efficiency offered by Arnold, which is focused on memory optimization, to be extended to software executable code.

The suggestion/motivation for doing so would have been derived from Arnold's suggestion of extending their optimization to numerous facets of programming (column 5, line 65-67):

- *as the computer program being generated as a result of the translation process, may be utilized using any number of representations, whether human or machine readable in nature.*

Therefore, it would have been obvious to combine Bates with Arnold to obtain the invention as specified in the instant claims.

Conclusion

11.   The prior art made of record and not relied upon is considered pertinent to applicant's

disclosure.

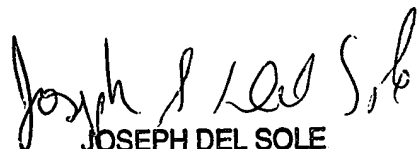Document Number: Name:

US-5,442,790 S   Nosenchuck, Daniel M.

US-5,481,708 A   Kukol, Peter

US-6,701,520 B1   Santosuosso et al.

US-2004/0019770   Kawahito, Motohiro

US-4,843,545   Kikuchi, Sumio

US-2004/0221281 A1   Suganuma, Toshio

US-5,581,696   Kolawa et al.

US-2005/0138611 A1   Inglis et al.

US-6,158,048   Lueh et al.

US-T097,200314   Coleman, Jr., Leslie Stinson

US-2004/0128660   Nair et al.

   Any inquiry concerning this communication or earlier communications from the examiner

should be directed to Evral Bodden whose telephone number is 571 272 3455. The examiner can

normally be reached on Monday to Friday, 8:30 to 5:00.

   If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor,

Joseph Del Sole can be reached on 5712721130. The fax phone number for the organization where

this application or proceeding is assigned is 571-273-8300.

   Information regarding the status of an application may be obtained from the Patent Application

Information Retrieval (PAIR) system. Status information for published applications may be obtained

from either Private PAIR or Public PAIR. Status information for unpublished applications is available

through Private PAIR only. For more information about the PAIR system, see http://pair-

direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the

Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a

USPTO Customer Service Representative or access to the automated information system, call 800-

786-9199 (IN USA OR CANADA) or 571-272-1000.

JOSEPH DEL SOLE
SUPERVISORY PATENT EXAMINER
4|10|07